Brigham Young University
BYU ScholarsArchive

Theses and Dissertations

2008-03-10

A BitTorrent Proxy

Robert Brent Larsen
Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Computer Sciences Commons

BYU ScholarsArchive Citation
Larsen, Robert Brent, "A BitTorrent Proxy" (2008). *Theses and Dissertations*. 1336.
https://scholarsarchive.byu.edu/etd/1336

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

A BITTORRENT PROXY

by

Robert B. Larsen

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

April 2008

Copyright © 2008 Robert B. Larsen

All Rights Reserved

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Robert B. Larsen

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Daniel Zappala, Chair
Date	Phillip Windley
Date	Dan Olsen

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Robert B. Larsen in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	Daniel Zappala Chair, Graduate Committee

Accepted for the Department

Date	Parris K. Egbert Graduate Coordinator

Accepted for the College

Date	Thomas W. Sederberg Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

A BITTORRENT PROXY

Robert B. Larsen

Department of Computer Science

Master of Science

BitTorrent is a peer-to-peer protocol useful for distributing large files over the Internet. Many organizations use BitTorrent to distribute their software in order to reduce client download time and reduce the load on their servers. While there is a lot of legitimate content available via BitTorrent, some organizations ban BitTorrent usage due to concerns over copyright infringement and the amount of bandwidth that peers can consume.

A BitTorrent proxy will allow organizations to control those risks and allow its members to use the BitTorrent protocol for approved uses. It will allow the organization to control the files that are downloaded and the total bandwidth that can be used, and it will eliminate redundant downloads by acting as a cache.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Zappala, for all the time and effort he spent helping me get this project done. He has been both patient and helpful, and has always been there to encourage me.

I appreciate all of my professors here at Brigham Young University who have dedicated themselves to provide an excellent education to myself and my fellow students.

I would also like to give a special thanks to my wife Julie and daughter Hailey for putting up with me through this process. It has taken a lot of time from them, but they have never waivered in their support.

Contents

List of Figures

Chapter 1

Introduction

BitTorrent is a peer-to-peer protocol for distributing large, popular content over the Internet. Peer-to-peer software enables users to download content from each other, avoiding the bottleneck of a central server. In BitTorrent, clients initially contact a central server to get a list of peers, then negotiate with the peers to download blocks of the file.

Organizations can receive significant benefits by distributing their content via BitTorrent. They can reduce the load on their servers and still provide customers with faster downloads by using torrents to distribute content. If a download is popular, a traditional web server can get bogged down with the many requests, and the customer experience suffers. With BitTorrent, clients can download popular files more quickly without taxing the main server.

Because of these benefits, there is a lot of useful, legal content available for distribution through the BitTorrent protocol. Many software companies and open source projects, including OpenOffice.org and Eclipse.org, distribute their software through torrents. In addition, most Linux distributions are available through torrents. Many artists who are interested in gaining a wider audience post their music and videos for downloads as well. These types of downloads are found on websites such as www.legaltorrents.com.

However, despite the benefits, many organizations are forced to ban the use of BitTorrent by its members for three main reasons. The first reason is because Bit-

Torrent can be used to download illegal content, including pirated software, movies [11], music, etc [5]. Several BitTorrent sites that provide content have been shut down because of copyright issues [6]. Users who download these files are violating copyright laws, and an organization can be held responsible for any copyright infringement by its members [9]. Vicarious infringement occurs whenever there is an infringement and a party who benefits from the infringement was also in a position to control the actions [13]. Employers and schools can be held responsible as vicarious infringers, and even if the organization is not ultimately held responsible, the legal costs and publicity are best avoided. Second, BitTorrent clients can consume a lot of bandwidth [7]. CacheLogic performed a study on their routers in 2004 and found that BitTorrent consumed up to 30% of the total bandwidth throughout their networks [2]. More recent studies have shown that peer-to-peer traffic still accounts for a large percentage of total bandwidth. Even in organizations that don't ban BitTorrent clients, administrators may still choose to limit the amount of bandwidth that BitTorrent clients can use [10]. Finally, organizations may choose to prohibit BitTorrent because of the large volume of objectionable content available through peer-to-peer software. Even if a file is legal and is not an undue burden on the network, it may not be suitable for the environment. Many sites provide pornographic or other content that the organization has an interest in eliminating. An additional problem with this material is that security can be compromised with the large amount of spy-ware and viruses associated with these types of files [8].

Our solution to balance legitimate organization concerns with the benefits of downloading content via BitTorrent is to create a BitTorrent proxy. This proxy is similar to a web caching proxy and provides a central location for controlling the use of BitTorrent. This centralized control allows an administrator to filter out copyrighted works and objectionable content. It also gives the benefit of reducing redundant downloads and the ability to control how much bandwidth can be used

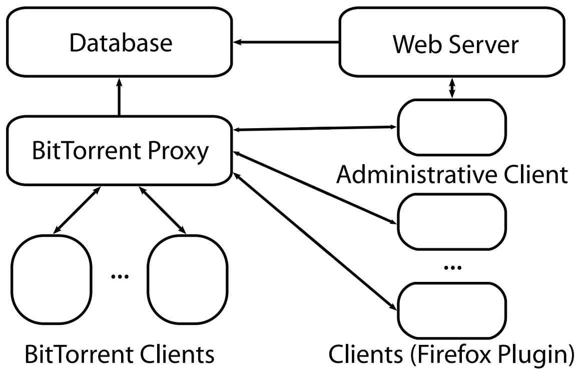


Figure 1.1: Architecture of the BitTorrent Proxy System: the directional arrows indicate communication between components.

for BitTorrent traffic. Users outside of the organization benefit as well by having a constant source for torrents the organization makes available.

The overall architecture of the BitTorrent Proxy, as shown in Figure 1.1, can be divided up into three basic components:

1. BitTorrent Proxy

   The BitTorrent Proxy receives download requests from the entire organization. When the proxy receives a download request, it checks a database to find the current status of the torrent. If the torrent is approved, the proxy initiates a download using a standard BitTorrent client. Once the download is complete, the proxy updates the database to reflect the new status. If the download would exceed bandwidth limits, the proxy adds the torrent to a download queue and the torrent is scheduled as bandwidth becomes available.

   The proxy responds to web clients with an HTML page that includes the current status of the torrent download, and provides a download link if applicable.

   The proxy also starts *BitTorrent seeds*, which allow torrent distribution when approved by the administrator. When a seed is authorized, the proxy immediately begins to upload the requested torrent. All uploads are restricted by a global upload bandwidth cap.

2. Web Administration

   The administration component provides the organization with the ability to control which files are allowed and the usage of organization resources. All modifications to the filtering rules and BitTorrent configuration are controlled with this component. The administrative component provides a web-based interface to control files available for download, files available for upload to other BitTorrent clients, and maximum bandwidth allotments.

   Administrators have the option of allowing single files, or entire sites through a whitelist. When the administrator approves a torrent request, it is recorded in the database.

   The administrative component also provides settings to control the maximum upload and download bandwidth usage of the BitTorrent client.

3. Firefox Extension

   The end-user's interface is a Firefox extension. When a user requests a torrent file, the extension redirects the request to the BitTorrent proxy. A simple web page is shown with the status of that file, whether it is approved or denied, and the download progress. The download progress automatically updates through AJAX. Once the file has been downloaded, the user can download the file through HTTP by clicking on a hyperlink.

We have developed a BitTorrent Proxy to provide this functionality to organizations. Our version is currently running on keystone.byu.edu, a server in the Internet Research Laboratory at Brigham Young University. This server also hosts the administration portion accessible at `http://keystone.byu.edu/~rlarsen/index.php`. This BitTorrent Proxy and administration have been tested and are available for general use.

Chapter 2

Background and Related Work

BitTorrent is a peer-to-peer protocol designed by Bram Cohen [4]. It is designed to facilitate the transfer of large files efficiently across the Internet. BitTorrent provides this functionality through torrent files, trackers, seeds, and peers.

There are several steps that need to be taken to share a file over BitTorrent. First, a user needs to split up the file into pieces and configure a tracker. The user places information describing how the file is split up in a .torrent file along with other information about the file. This .torrent file also contains the URL of the tracker. The user then makes the .torrent file available for download by placing it on a web server.

In order to start downloading a file, BitTorrent clients download the .torrent file and use the information to connect to the tracker. The tracker is responsible for helping the peers to locate each other. The tracker will give a list of current peers to anyone who contacts it; once a client receives this list, the transfer can begin taking place.

Peers can download pieces of the file from each other and from seeds. A seed is a peer that has a complete copy of the file, either because it was started by the person sharing the file or because it downloaded a complete copy. The seed will share indiscriminately with other clients, and should not ever refuse a request.

Peers that are not seeds use tit-for-tat incentives to dictate fairness in client usage. If a client grants a request from a peer and sends a piece of the file, it will expect that client to provide a piece of the file in return. If it does not receive a piece before the next request is made, the client will choke that peer, meaning that future requests will be ignored until the requester has uploaded a needed piece of the file. This ensures that clients do not just leech off of other peers, but actively participate in the torrent. This also has the effect of making the downloads more efficient for everyone. Clients will typically only unchoke the fastest 2 to 4 peers at any given time. Occasionally, the client will optimistically unchoke a new peer to try to find a faster peer. This process makes it advantageous to all clients to upload as fast as they can so that they will be unchoked by other peers. Generally, the faster a client gives data the faster it will receive data.

Once a client has downloaded the entire file, it may become a seed. It is expected that clients stay in a torrent for a time after finishing the download, but it is not required. Seeds no longer need any pieces, so they never choke their peers.

2.1 Web caching proxies

Another way to help web clients download faster and reduce load on web servers is a web caching proxy. Web caching proxies accept requests from clients and service them either internally from the cache or by passing them to other servers. A caching proxy must decide if the cached content is up-to-date and what content to discard when the cache becomes full. As long as the cache is current and the requested information is located in the cache, the proxy will act as though it is the target web server. Otherwise, the proxy contacts the target web server and forwards the response to the client as well as adding the new content to its cache.

Web caching proxies provide a faster download for clients. Commonly requested content is located on the local network in the cache, so clients do not need to wait for slow WAN links or high-latency connections. The local network usually has much faster speeds and more available bandwidth.

Web servers also benefit from web caching proxies. The proxies lower the load on the server, since many clients can be serviced with a single response to the proxy. Since there are fewer direct requests, the server does not need to work as hard and uses less bandwidth overall.

Web servers cannot control or predict the behavior of clients, however. The web server only benefits when multiple requests are made from the same network through a proxy server. Even with proxies taking some of the load, a small number of extra clients can easily overload a web server with low bandwidth or an already high load. BitTorrent files are typically very large, and would not normally be cached by the proxies. In this case, the proxies offer little help, since they will request the file multiple times anyway.

Squid is one implementation of a web caching proxy [3]. It is freely available as open source and provided with many Linux distributions.

2.2 Content filtering

Content filtering programs grant or deny access to resources based on the content of that resource. The filter uses rules defined by the organization to allow only approved content to be downloaded and viewed by users. Content filters must scan the application level HTTP messages for prohibited content, and typically replace offending pages with a modified HTML document. This modification can either be a complete denial, or just removing content from the page.

One way to filter content is by using the URL of the requested material. Although this requires someone to make a list of URLs, once this list is available filtering is easy. The content filter just matches the requested URL to the list to decide whether or not to deny the content. These lists can come as either white lists or black lists. If the URL matches an entry in the white list, the content is deemed acceptable and forwarded to the requesting client. If the URL matches a black list entry, the content is blocked, and a modified page is forwarded to the client.

URLs that do not match a list must be filtered another way. In addition to URL matching, DansGuardian also uses a system of keyword matching to filter web pages [1]. A keyword can be a single word, or a group of words close together. Each keyword is assigned a value based on how offensive it is. These values can be positive or negative. The filtering process consists of adding up the values of any occurrences of keywords in the page. The site administrator picks a certain total that is acceptable, and anything with a higher total is blocked. This method of filtering allows much more flexibility for organizations.

Neither of these filtering methods is successful at blocking offensive images or video. PornSeer Pro is an example of a different kind of filtering software that actually "looks" at images to decide if they are acceptable or not [14]. These software filters try to match pieces of images to human body parts or other objectionable content. If a match is detected, the image is replaced or edited before the page is forwarded to the client.

Email filters often use a Bayesian classification system to filter junk emails. With this method, the message is parsed and reduced to a list of words. Certain words, such as free, and phrases can indicate that a message is spam. Bayesian filtering attempts to learn the probabilities of whether or not a message is spam by using a set of email messages. Using this set of messages, a probability is associated with words, and the overall probability is calculated by the presence of interesting words [12].

Chapter 3

BitTorrent Proxy

The BitTorrent Proxy consists of two main responsibilities. First, it handles web requests from users, and second, it is responsible for launching and controlling a BitTorrent client.

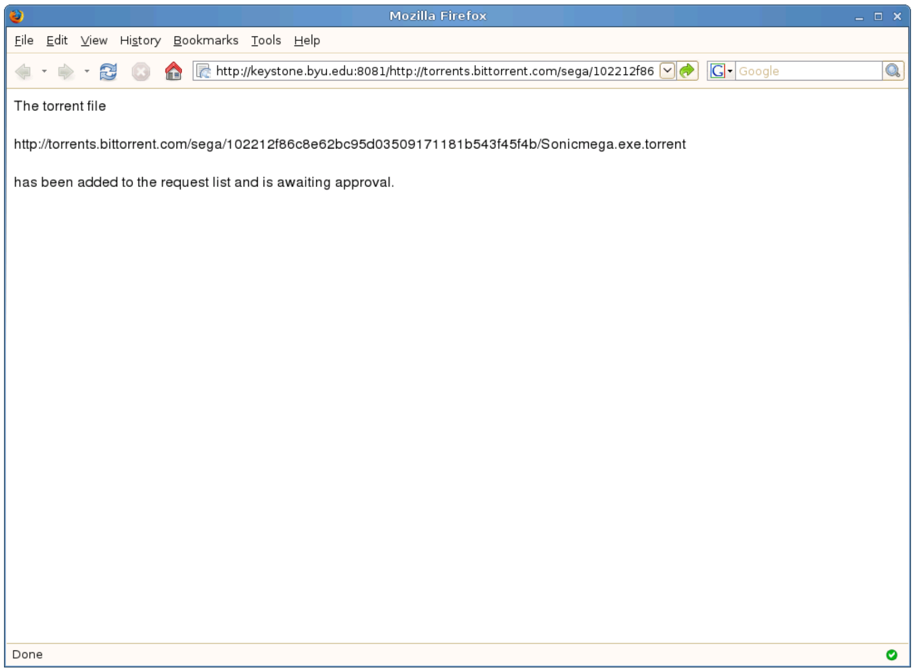
3.1 Web Requests

When the proxy software is loaded, it first initiates a simple web server to handle requests for torrent downloads. This server opens a port (8081 by default) and begins to listen for incoming requests.

An incoming request is a simple GET request where the URL consists of the proxy server's address followed by the full URL for the desired torrent file. For those using the Firefox extension described in the introduction and discussed later, this request is formatted automatically for links to torrent files. A typical request has the form "`http://proxy:8081/http://www.foo.com/file.torrent`".

When a request is received, the proxy server parses the request to read the filename. If the requested torrent file is already in the database, it returns a response page containing the current status of the download. There are four possible status messages:

1. If a requested torrent is not yet approved and is not from a whitelisted site, the message informs the user that the torrent has not been approved yet, but might


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://keystone.byu.edu:8081/http://torrents.bittorrent.com/sega/102212f86
Google
Done

The torrent file

http://torrents.bittorrent.com/sega/102212f86c8e62bc95d03509171181b543f45f4b/Sonicmega.exe.torrent

has been added to the request list and is awaiting approval.

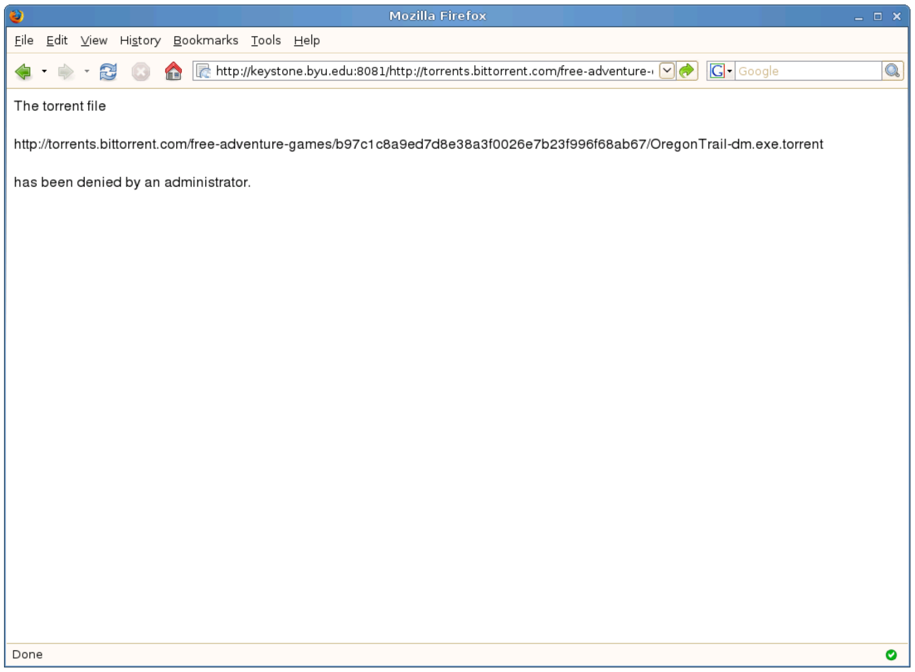
Figure 3.1: Torrent status: pending approval

be downloaded in the future.(Figure 3.1) In the future processes might be added to automatically approve torrents based on specific criteria, but currently all torrents from sites that have not been whitelisted must me manually approved by an administrator.

2. If a torrent request has previously been denied, the message indicates that the torrent has been denied and will not be downloaded.(Figure 3.2)

3. If the torrent is currently being downloaded, the message shows the percent of the download that is completed.(Figure 3.3) The percentage is automatically updated through AJAX requests as the BitTorrent client makes progress downloading the file.

4. If the torrent file has been fully downloaded and is available on the proxy server, the message provides a hyperlink to download the file from the proxy server using HTTP.(Figure 3.4)

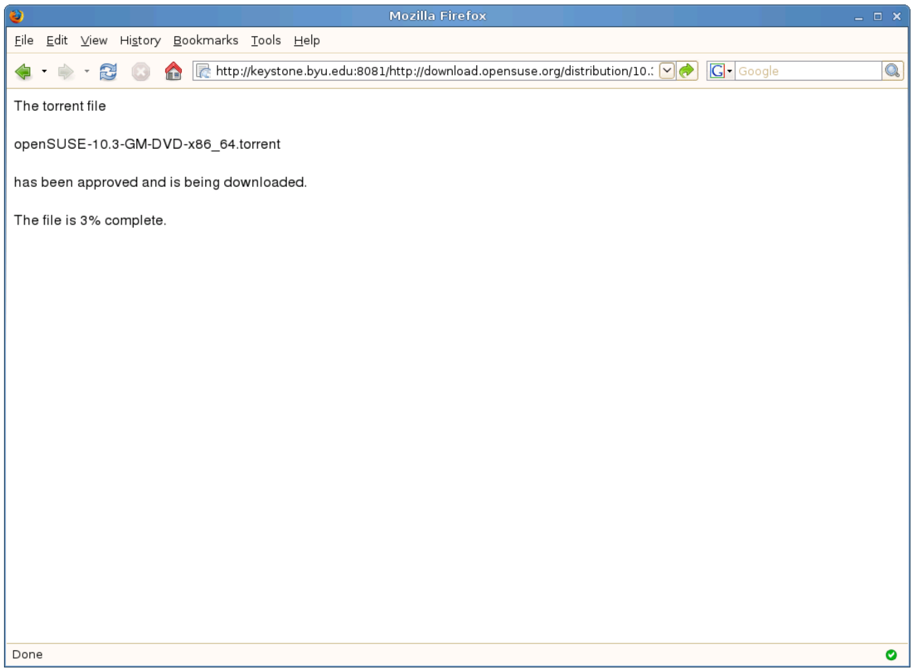

Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://keystone.byu.edu:8081/http://torrents.bittorrent.com/free-adventure-
Google
Done

The torrent file

http://torrents.bittorrent.com/free-adventure-games/b97c1c8a9ed7d8e38a3f0026e7b23f996f68ab67/OregonTrail-dm.exe.torrent

has been denied by an administrator.

Figure 3.2: Torrent status: denied


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://keystone.byu.edu:8081/http://download.opensuse.org/distribution/10.:
Google
Done

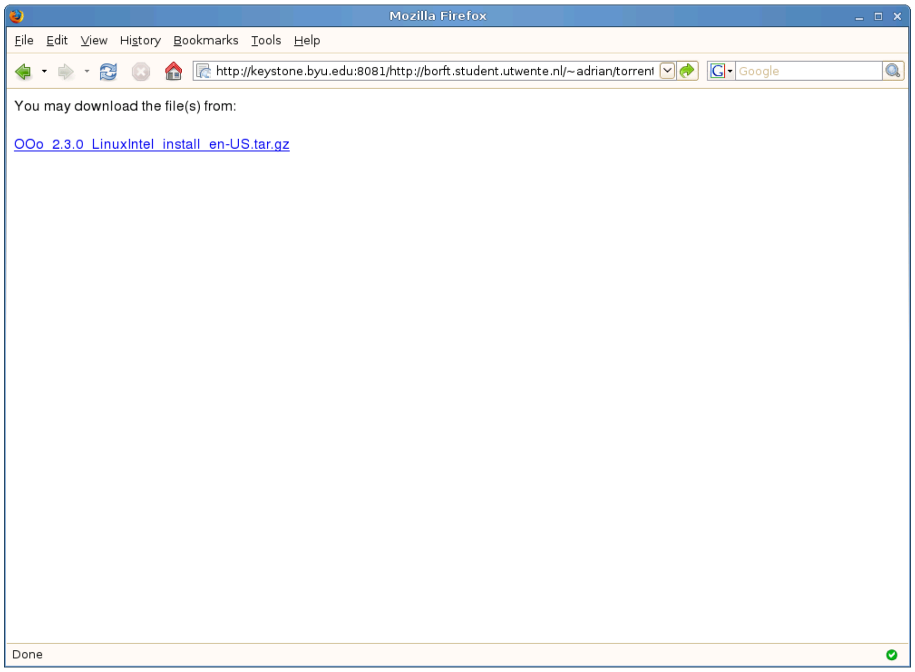
The torrent file

openSUSE-10.3-GM-DVD-x86_64.torrent

has been approved and is being downloaded.

The file is 3% complete.

Figure 3.3: Torrent status: downloading


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://keystone.byu.edu:8081/http://borft.student.utwente.nl/~adrian/torrent
Google
Done

You may download the file(s) from:

OOo_2.3.0_LinuxIntel_install_en-US.tar.gz

Figure 3.4: Torrent status: available

If the requested file is not in the database, the proxy downloads the torrent information using the provided URL and stores the requested torrent as a new entry. If the URL begins with one of the whitelist strings, the torrent is approved immediately, otherwise the status is set to await approval by the administrator. One of the previously listed messages is returned to the user to show the current status of the torrent request.

3.2 Torrent Downloads

Once a torrent is approved, it must be downloaded before it is available for users. The BitTorrent protocol is well defined and there are many implementations that we can choose from.

Our BitTorrent Proxy was developed around the Python implementation of the version 5.0.7 BitTorrent client written by Bram Cohen and available at www.bittorrent.com. Our simplified client for the BitTorrent Proxy is also written in Python and depends upon the MultiTorrent library.

A MultiTorrent is a Python class that can contain many individual torrents at the same time. It is defined in the MultiTorrent.py source file distributed with the BitTorrent client from bittorrent.com. The following code demonstrates how to create a MultiTorrent and start a torrent.with a URL located at torrentUrl.

```

from BitTorrent.defaultargs import get_defaults
from BitTorrent import GetTorrent
from BTL.bencode import bdecode
from BTL.ConvertedMetainfo import ConvertedMetainfo
from BitTorrent.RawServer_twisted import RawServer
from BitTorrent.MultiTorrent import MultiTorrent

self.config, args = configfile.parse_configuration_and_args( \
    get_defaults('bittorrent-console'), 'bittorrent-console')
self.rawserver = RawServer(self.config)
self.multitorrent = MultiTorrent(self.config, self.rawserver,\
                              '.', resume_from_torrent_config=False)
info = GetTorrent.get_url(torrentUrl)
metainfo = ConvertedMetainfo(bdecode(info))
if metainfo is not None:
    infoHash = metainfo.infohash
    if self.downloads[torrentUrl].is_initialized():
        self.multitorrent.start_torrent(metainfo.infohash)
```

Other torrents can easily be added to this MultiTorrent by reading the metainfo from the torrent file and then calling start_torrent().

Configuration options for the torrent can either be specified at startup or set during runtime by calling the MultiTorrent.set_option(option, value) function. These options are shared by each torrent in the MultiTorrent. When an individual torrent is finished it will call the finished() function of the calling class, allowing other tasks to be executed upon completion.

The proxy utilizes a MultiTorrent to actually download the torrents. The MultiTorrent calls a function named UpdateTorrentList every five seconds to find new work and update the status of all current torrents. UpdateTorrentList makes a database query to check for any torrents that have been approved, but are not fully downloaded. If there is sufficient bandwidth to download the file, it is added to the MultiTorrent. UpdateTorrentList also queries the MultiTorrent for the current status of its torrents and updates the database with the completion percentage.

3.3 Torrent Uploads

BitTorrent seeds are simply clients that have a complete copy of the entire file. Seeding a file just requires the torrent to be added to the MultiTorrent, exactly as described above. Once the torrent is started, it will respond to any download requests.

The UpdateTorrentList function described above also queries the database for seeds approved by the administrator. Any approved seeds are added to the MultiTorrent and the bandwidth is controlled by the maximum upload bandwidth set on the client.

3.4 Torrent Scheduling

Torrents are scheduled to upload or download in a first come first served fashion. The BitTorrent client checks the database for incomplete, approved torrents at specific time intervals. Any incomplete torrents are added to the current torrent list, as long as there is sufficient bandwidth free. If there is not enough available bandwidth, the torrent will not be added to the download list.

Bandwidth availability is determined by adding the current BitTorrent download bandwidth and a simple estimation of bandwidth requirements for a torrent. The proxy assigns a bandwidth requirement of 500 Kbps to each new torrent. The first new torrent is added to the download list if there is any available bandwidth. Subsequent torrents are only added to the list if the current bandwidth plus the bandwidth requirements for the new torrents already added does not exceed the maximum bandwidth. If there are four new torrents, but only 750 Kbps of bandwidth not currently being used, only two torrents will be added to the client's download list.

Torrents that have been approved as seeds will be added regardless of the current upload bandwidth. The maximum upload bandwidth is controlled by the administrator and respected no matter how many torrents are being seeded. The administrator is responsible to control the number of total seeds to avoid possible performance issues for users downloading those torrents.

3.5 Testing

The component requires testing to ensure that the proper responses are sent for each request. There are five basic tests that need to be run. Each of these tests are run both manually and through a BASH script calling the wget command. The script can be run on various machines and/or multiple times providing a larger load; it uses grep to verify that the responses are correct.

1. Pending

   A request is made for a new torrent or one that has not been approved yet. The proper response is found in Figure 3.1 above and should state that the torrent has not been approved yet.

2. Denied

   A request is made for a torrent that has been denied by the administrator. The response to this request is found in Figure 3.2 and needs to state that the torrent has been denied by an administrator.

3. Downloading

   A request is made for a torrent that is currently in the process of downloading. The progress should be shown as depicted in Figure 3.3.

4. Available

   A request is made for a torrent that is fully downloaded and available for users. A link must be provided for users to download the file as shown in Figure 3.4.

5. Uploading

   A BitTorrent client makes a request for a torrent file that is currently being seeded. The seed should respond and upload blocks to the requesting client.

Chapter 4

Web Administration

The web interface for the BitTorrent Proxy is written in PHP and Javascript. It provides an interface for users to browse torrents and administrators to control content and parameters. It also allows users to register and request a torrent seed.

Authentication to the web services is session-based, with users contained in the database. The users table, as well as the other database tables, is described in the appendix.

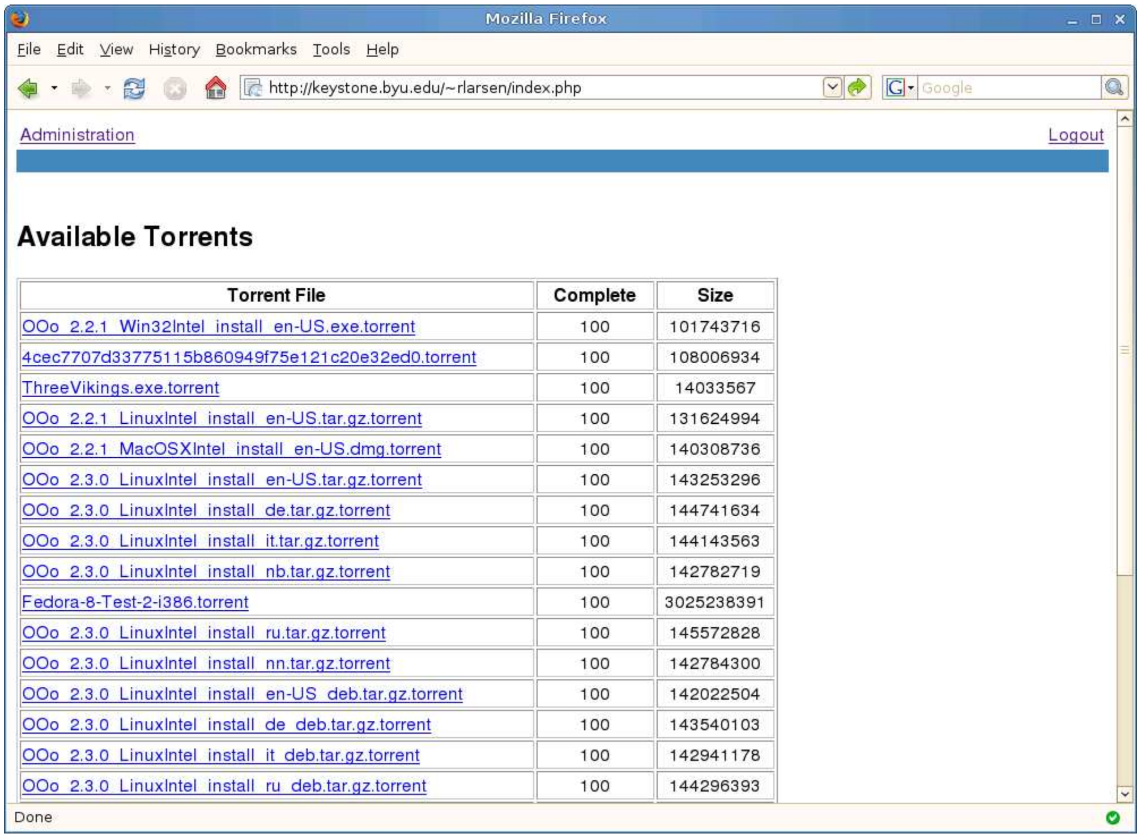
4.1 Usage

When any user accesses the web interface with a browser, it will display a list of torrents as shown in Figure 4.1. Clicking on a torrent in that list will display the informational page shown in Figure 4.2 about the torrent with a link for download if available.

4.1.1 Administration

The primary role of the web interface is administration. On the administration page, there are four tabs: Torrents, Whitelist, Seeds, and Status.

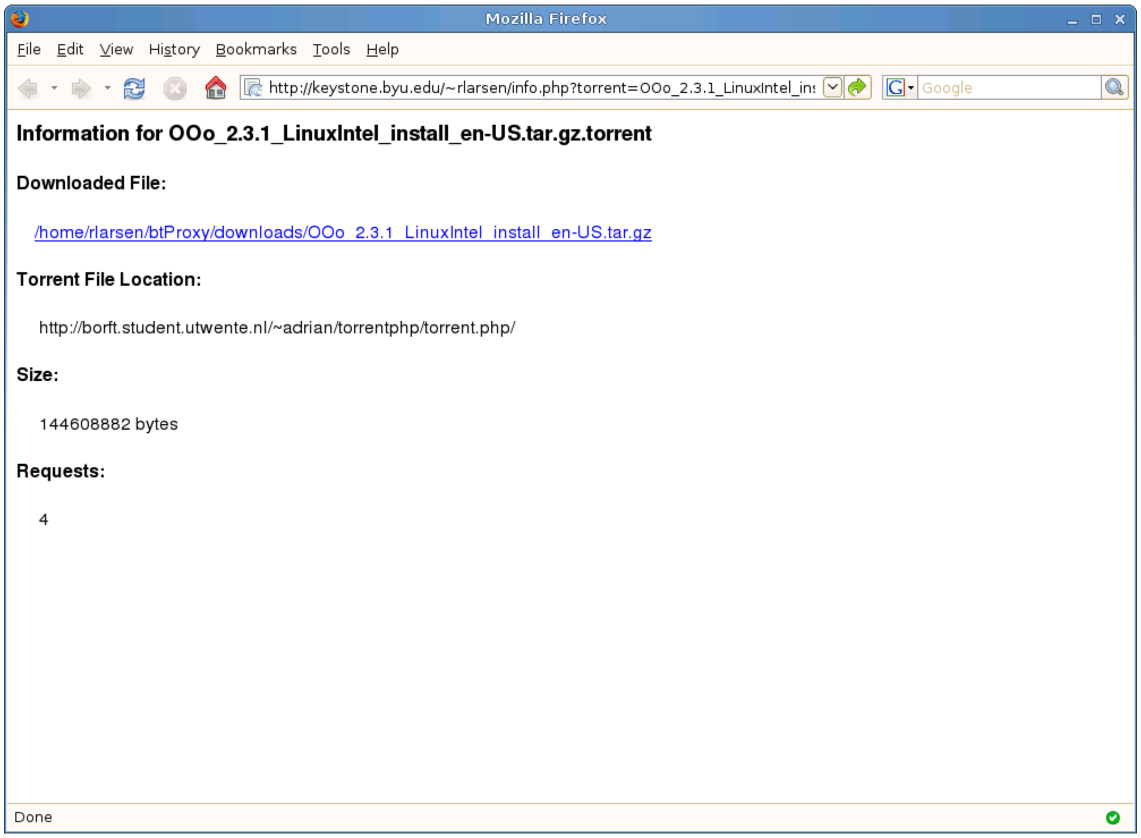
The Torrents tab, shown in Figure 4.3, allows the administrator to approve or deny torrents. All torrents requests are listed under this tab and any torrent status can be updated, regardless of the previous status. Previously denied torrents can be


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://keystone.byu.edu/~rlarsen/index.php
Google
Administration
Logout
Done

Available Torrents

Torrent File	Complete	Size
OOo_2.2.1_Win32Intel_install_en-US.exe.torrent	100	101743716
4cec7707d33775115b860949f75e121c20e32ed0.torrent	100	108006934
ThreeVikings.exe.torrent	100	14033567
OOo_2.2.1_LinuxIntel_install_en-US.tar.gz.torrent	100	131624994
OOo_2.2.1_MacOSXIntel_install_en-US.dmg.torrent	100	140308736
OOo_2.3.0_LinuxIntel_install_en-US.tar.gz.torrent	100	143253296
OOo_2.3.0_LinuxIntel_install_de.tar.gz.torrent	100	144741634
OOo_2.3.0_LinuxIntel_install_it.tar.gz.torrent	100	144143563
OOo_2.3.0_LinuxIntel_install_nb.tar.gz.torrent	100	142782719
Fedora-8-Test-2-i386.torrent	100	3025238391
OOo_2.3.0_LinuxIntel_install_ru.tar.gz.torrent	100	145572828
OOo_2.3.0_LinuxIntel_install_nn.tar.gz.torrent	100	142784300
OOo_2.3.0_LinuxIntel_install_en-US_deb.tar.gz.torrent	100	142022504
OOo_2.3.0_LinuxIntel_install_de_deb.tar.gz.torrent	100	143540103
OOo_2.3.0_LinuxIntel_install_it_deb.tar.gz.torrent	100	142941178
OOo_2.3.0_LinuxIntel_install_ru_deb.tar.gz.torrent	100	144296393

Figure 4.1: Torrent listing


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://keystone.byu.edu/~rlarsen/info.php?torrent=OOo_2.3.1_LinuxIntel_ins
Google
Done

Information for OOo_2.3.1_LinuxIntel_install_en-US.tar.gz.torrent

Downloaded File:

/home/rlarsen/btProxy/downloads/OOo_2.3.1_LinuxIntel_install_en-US.tar.gz

Torrent File Location:

http://borft.student.utwente.nl/~adrian/torrentphp/torrent.php/

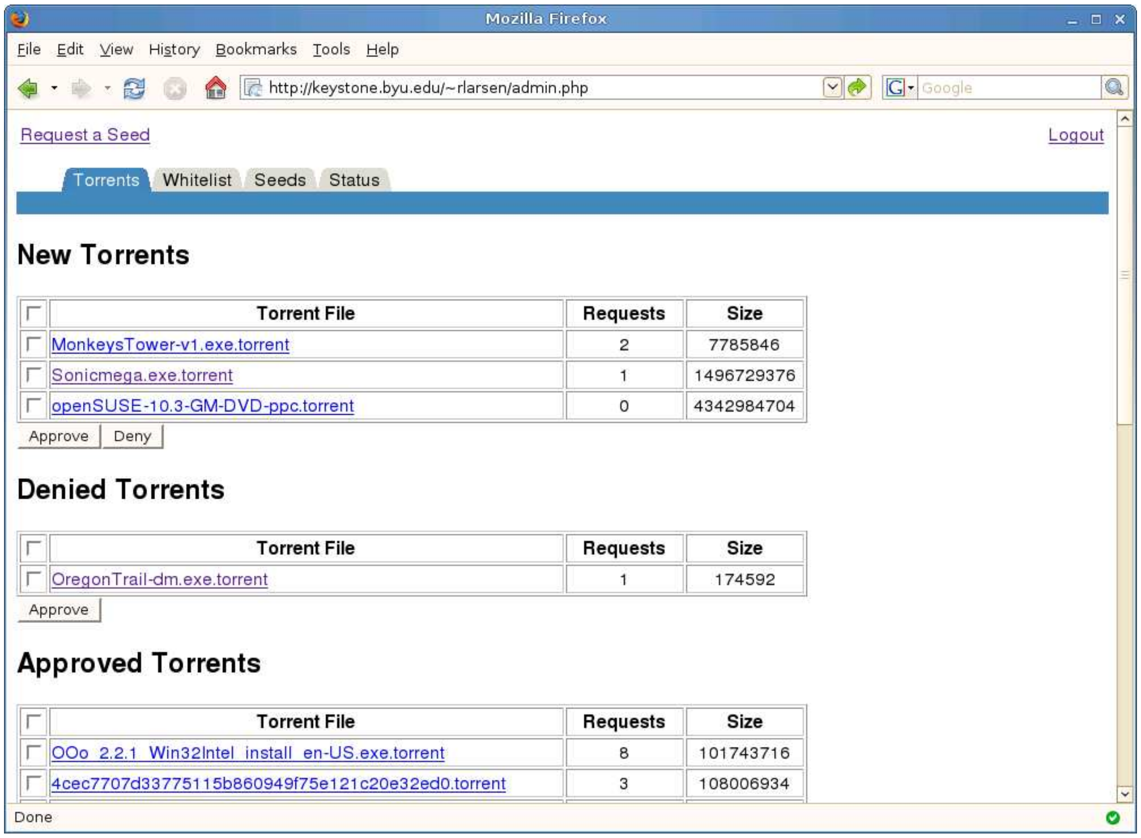
Size:

144608882 bytes

Requests:

4

Figure 4.2: Torrent information screen


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://keystone.byu.edu/~rlarsen/admin.php
Google
Request a Seed
Logout
Torrents
Whitelist
Seeds
Status
Done

New Torrents

☐	Torrent File	Requests	Size
☐	MonkeysTower-v1.exe.torrent	2	7785846
☐	Sonicmega.exe.torrent	1	1496729376
☐	openSUSE-10.3-GM-DVD-ppc.torrent	0	4342984704

Approve Deny

Denied Torrents

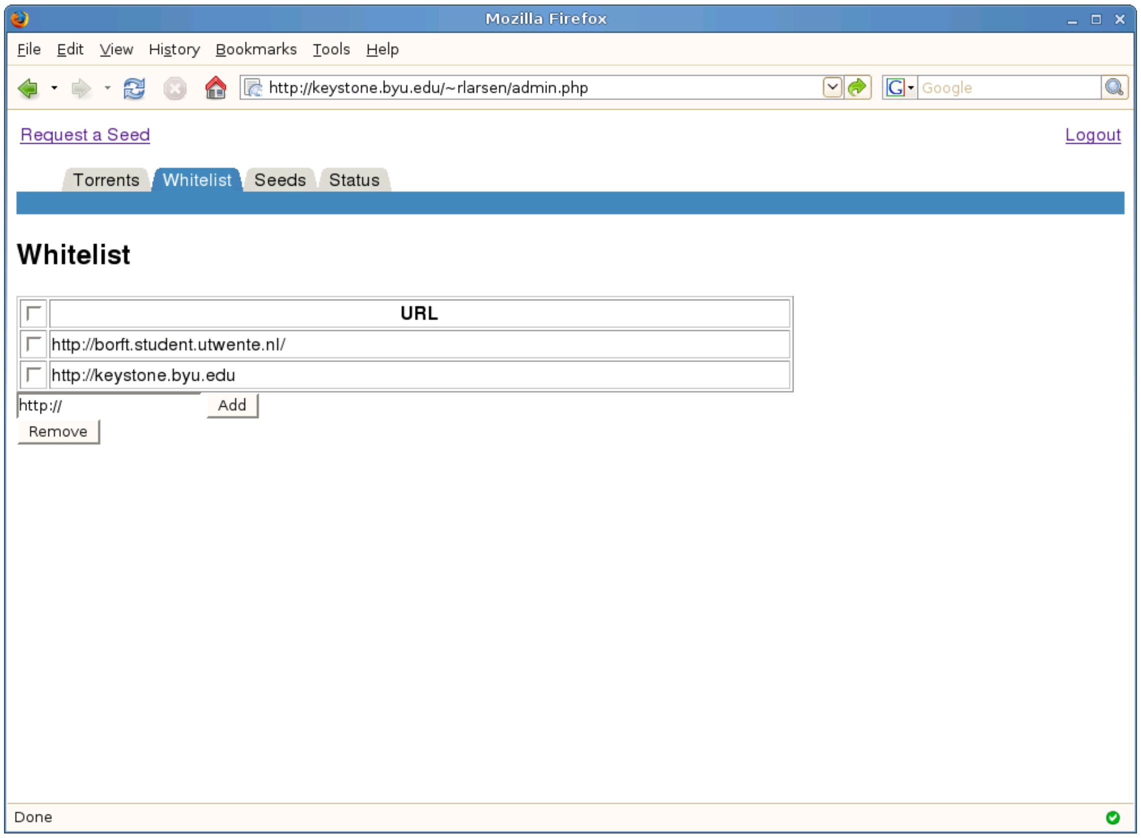
☐	Torrent File	Requests	Size
☐	OregonTrail-dm.exe.torrent	1	174592

Approve

Approved Torrents

☐	Torrent File	Requests	Size
☐	OOo_2.2.1_Win32Intel_install_en-US.exe.torrent	8	101743716
☐	4cec7707d33775115b860949f75e121c20e32ed0.torrent	3	108006934

Figure 4.3: Administration - Torrents tab


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://keystone.byu.edu/~rlarsen/admin.php
Google
Request a Seed
Logout
Torrents
Whitelist
Seeds
Status

Whitelist

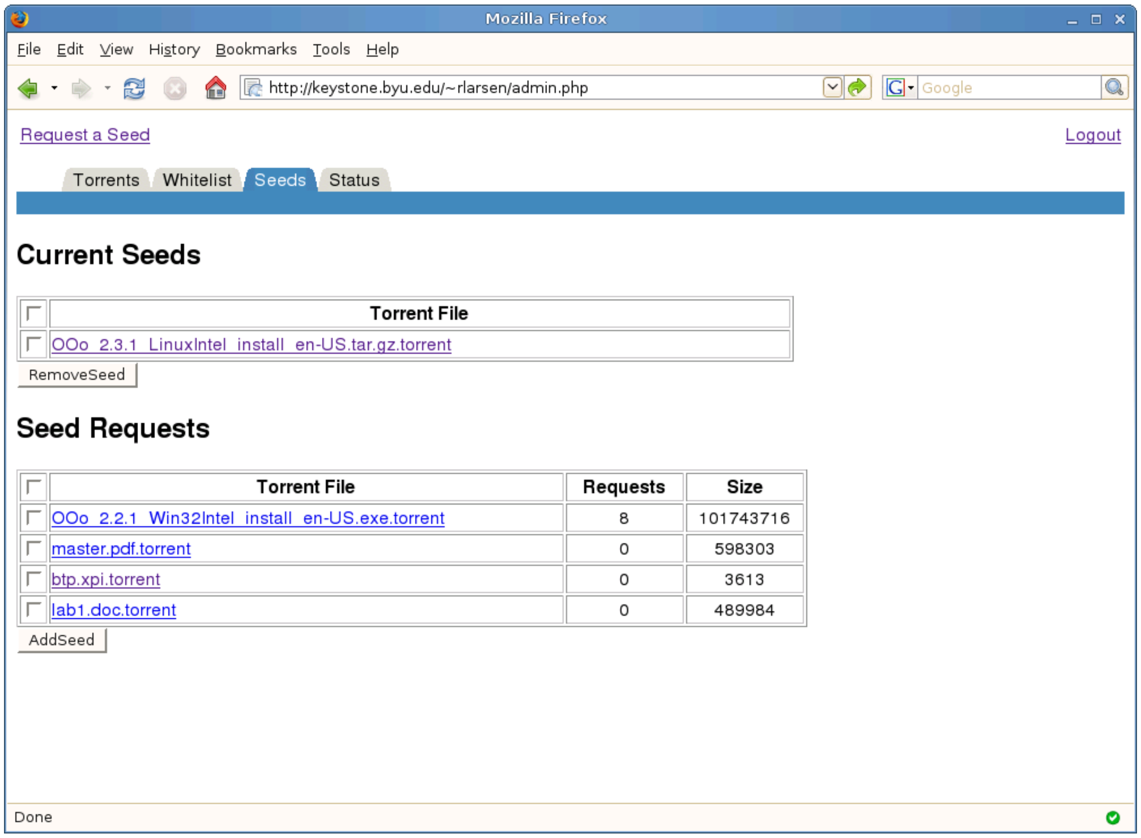
	URL
☐	http://borft.student.utwente.nl/
☐	http://keystone.byu.edu

Figure 4.4: Administration - Whitelist tab

approved and vice versa. Administrators can simply select multiple torrents and then click the approve or deny buttons to update the lists. Any torrents that are approved but not downloaded will be detected by the proxy and downloads will begin as soon as possible.

The Whitelist tab, shown in Figure 4.4, contains a list of all pre-approved sites for torrent files. Any request that begins with a string listed here will be automatically approved instead of waiting for approval. To add a site to the whitelist, the administrator simply types the URL beginning and clicks the add button. To remove a site, select the desired URL and click the remove button.

The Seeds tab (Figure 4.5) permits the administrator to control the torrents that the proxy will seed. The Current Seeds list contains all of the currently allowed seeds, and the Seed Requests list shows all of the seeds that have been requested by


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://keystone.byu.edu/~rlarsen/admin.php
Google
Request a Seed
Logout
Torrents Whitelist Seeds Status
RemoveSeed
AddSeed
Done

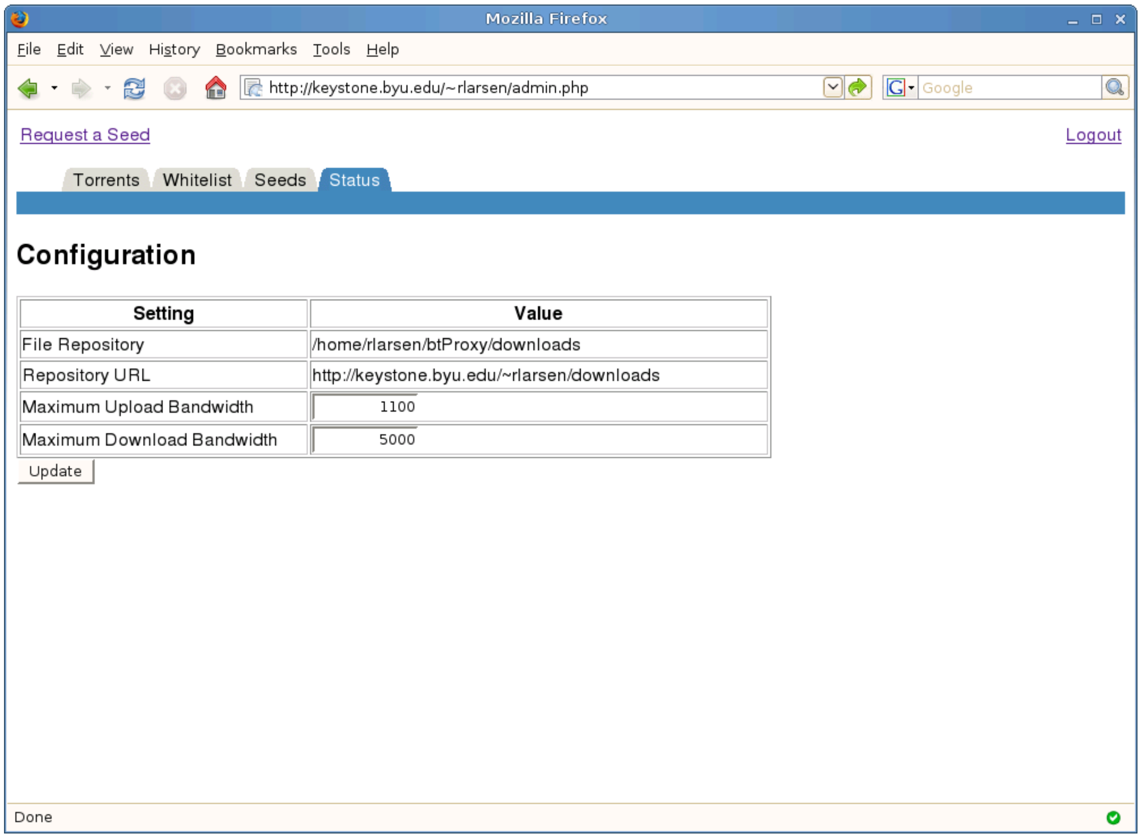
Current Seeds

	Torrent File
☐	OOo_2.3.1_LinuxIntel_install_en-US.tar.gz.torrent

Seed Requests

	Torrent File	Requests	Size
☐	OOo_2.2.1_Win32Intel_install_en-US.exe.torrent	8	101743716
☐	master.pdf.torrent	0	598303
☐	btp.xpi.torrent	0	3613
☐	lab1.doc.torrent	0	489984

Figure 4.5: Administration - Seeds tab


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://keystone.byu.edu/~rlarsen/admin.php
Google
Request a Seed
Logout
Torrents
Whitelist
Seeds
Status
Update
Done

Configuration

Setting	Value
File Repository	/home/rlarsen/btProxy/downloads
Repository URL	http://keystone.byu.edu/~rlarsen/downloads
Maximum Upload Bandwidth	1100
Maximum Download Bandwidth	5000

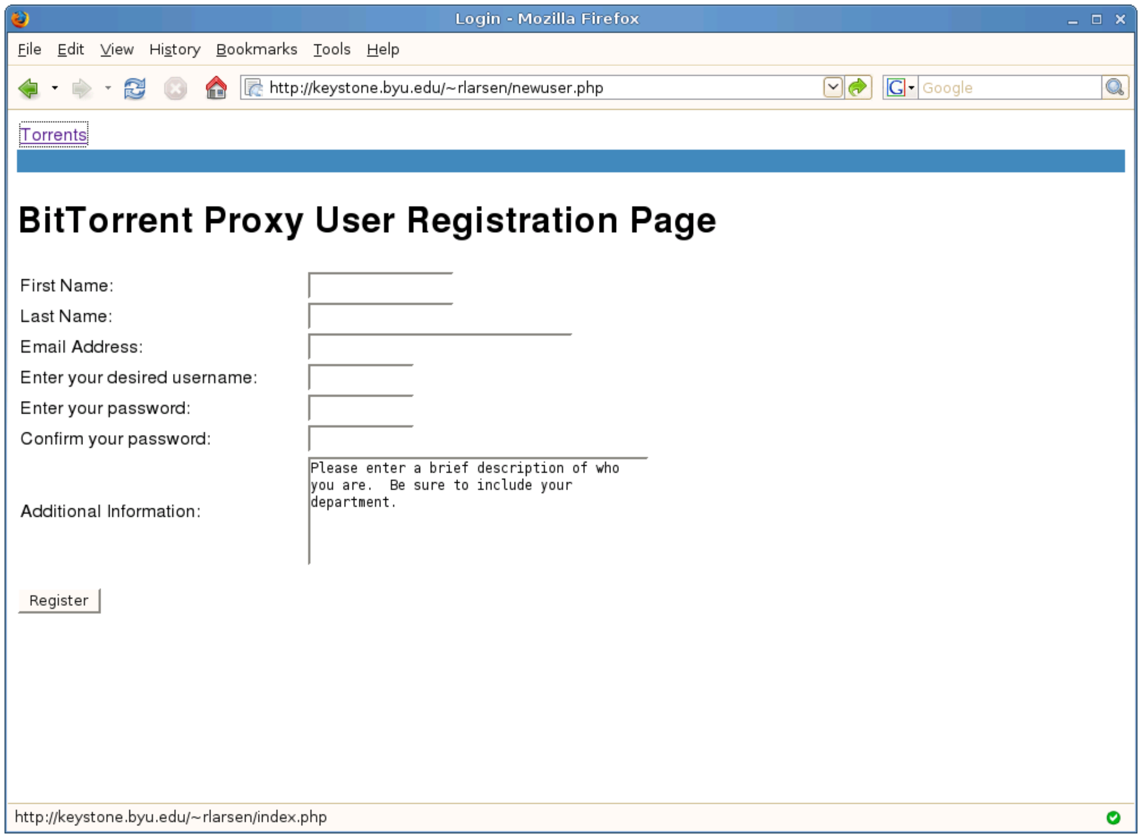
Figure 4.6: Administration - Status tab

users. To add or remove seeds from the Current Seeds list, select the torrent and click the appropriate button.

The Status tab, displayed in Figure 4.6, shows the current status and parameters of the proxy service. The maximum upload and download speeds can be updated and the current status of the services is displayed.

4.1.2 Registered Users

Users that have registered with the site can also make requests for the proxy to seed torrents. In order to register, the user must click on the registration link and fill out the form shown in Figure 4.7. This form collects information about the new user and stores the information in the database.


Login - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://keystone.byu.edu/~rlarsen/newuser.php
Google
Torrents

BitTorrent Proxy User Registration Page

First Name:
Last Name:
Email Address:
Enter your desired username:
Enter your password:
Confirm your password:
Additional Information: Please enter a brief description of who you are. Be sure to include your department.

Register

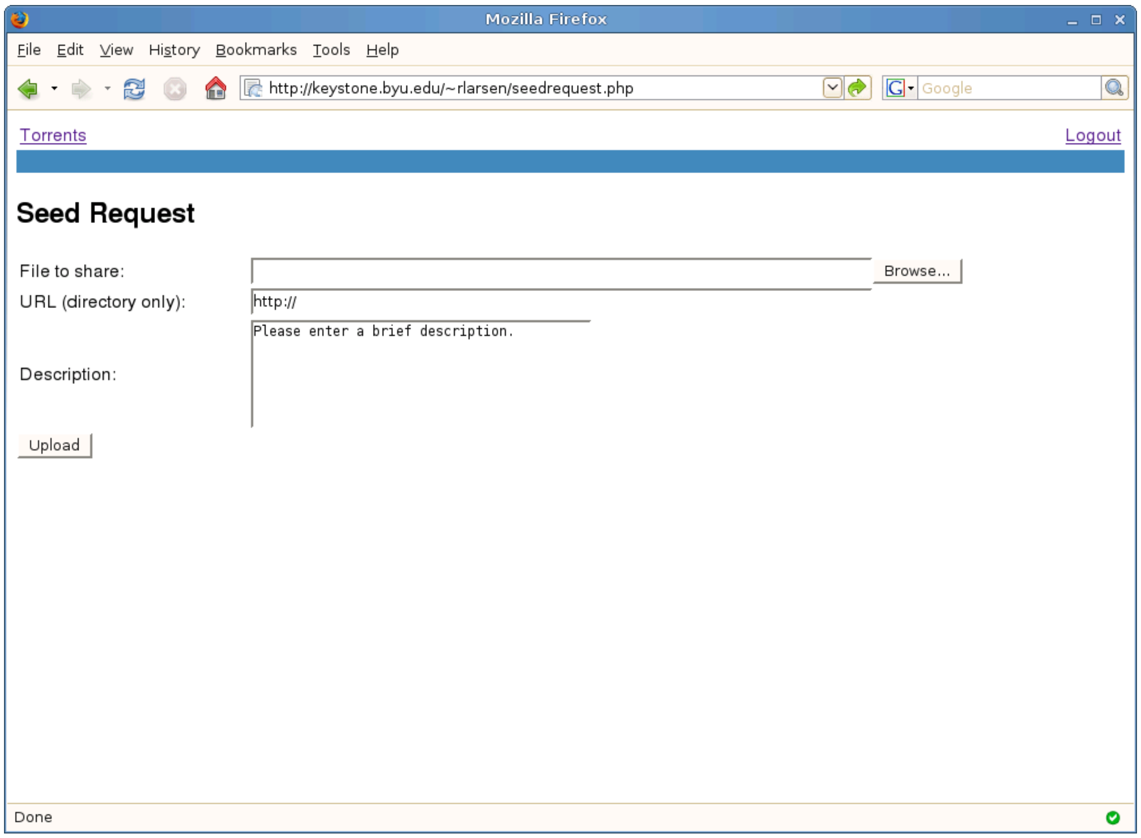
http://keystone.byu.edu/~rlarsen/index.php

Figure 4.7: User registration screen

After logging in, the user can access the page shown in Figure 4.8, browse to a file on their computer and upload that file to the proxy. The proxy will create a torrent file and make that file available to the user for download. Users will be responsible for sharing the .torrent file on a web server and should update the URL whenever it changes by requesting a seed for the same file again.

4.2 Testing

The Web Administration is a user interface so all testing is done manually to verify that pages are displayed properly to the user. Tests need to be run for the authentication component, including login and logout, and for the functionality for each type of user.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://keystone.byu.edu/~rlarsen/seedrequest.php
Google
Torrents
Logout
Browse...
Upload
Done

Seed Request

File to share:	
URL (directory only):	http://
Description:	Please enter a brief description.

Figure 4.8: Torrent request screen

4.2.1 Authentication

Login

Failed Login	Login with an invalid user. The authentication should fail and non-registered user status is assigned. The user doesn't have rights to request seeds or do administrative functions.
Registered User Login	Login with a valid user. The authentication should succeed and registered user status is assigned. The user has rights to request a seed, but can not perform administrative functions.
Administrative User Login	Login with an administrative user. The authentication should succeed and administrative status is assigned. The user has rights to perform all administrative functions, but can not request seeds.

Logout

Logout at any time. The authenticated session is cleared and the user is assigned non-registered status. The user should have no rights other than to view the torrent list.

4.2.2 User functions

View and download a torrent	All users whether authenticated or not should be able to view torrent information and download completed torrents.
Request a seed	Any registered user should be able to request a seed when logged in.
Approve a torrent	Administrative users should be able to approve a torrent.
Deny a torrent	Administrative users should be able to deny a torrent.
Add a URL to the whitelist	Administrative users should be able to add a URL to the whitelist.
Remove a URL from the whitelist	Administrative users should be able to remove a URL from the whitelist.
Approve a seed request	Administrative users should be able to approve a seed request.
Remove a seed request from the current list	Administrative users should be able to stop seeding a torrent.
Update the upload rate	Administrative users should be able to update the upload rate for the BitTorrent client.
Update the download rate	Administrative users should be able to update the download rate for the BitTorrent client.

Chapter 5

Firefox Extension

The BitTorrent Proxy Firefox extension is written in Javascript. The user interface consists of pages designed in XUL.

The key functionality of the extension is provided by the scanLinks function. When a new page loads, the onLoad function calls scanLinks, which checks each link for a .torrent ending and redirects any matching links to the specified proxy server and port. The scanLinks function is implemented as follows:

```
function scanLinks(e) {
    var doc = e.originalTarget;
    var links = doc.links;
    btp_init();
    var btpServer = nsIPrefBranchObj.getCharPref("btpProxyServer");
    var btpPort = nsIPrefBranchObj.getCharPref("btpProxyPort");
    for (var i=0; i<links.length; i++)
    {
        if ((links[i].href.match(/^http.*\.torrent$/))
                    && (links[i].href.match(btpServer) != btpServer))
            links[i].href=\
             "http://"+btpServer+":"+btpPort+"/"+links[i].href;
    }
}
```

5.1 Usage

While the BitTorrent extension is active, it will rewrite all links to .torrent files and redirect them to the BitTorrent Proxy server. Sites that use Javascript or other methods to deliver the .torrent files may not work properly.

If the torrent link points to a file servered by the BitTorrent Proxy web server, the link will not be rewritten. This is to allow users to download torrent files for files where they have requested a seed.

5.2 Testing

The BitTorrent Firefox extension was tested manually. After installation and configuration, a page with various .torrent file links and links with 'torrent' in the name were loaded. Testing passed when all links were properly rewritten.

Chapter 6

Future Work

Our BitTorrent Proxy is robust and functional, but by no means complete. There are several features or improvements that could be added to increase the usability and functionality of the proxy. Following is a brief description of possible improvements and additions which could be considered for future work.

6.1 Space Management

Even though hard drive space is becoming cheaper and more available, it is still important to conserve disk space where possible. As drives become larger, file sizes increase as well. The current BitTorrent proxy doesn't have any mechanism to control disk usage. Files will be downloaded and stored as long as there is still space to store them, and the files are never cleaned up or removed. An algorithm could be developed to delete files as they are getting older or aren't requested often. This process would conserve space on the server and still allow popular files to be stored for quick downloads..

6.2 Web Interface

The current web face is very simple and has very few graphics. Some work could be done to create a theme for the web interface to increase usability and improve the customer experience.

It might also be useful to provide users with statistics about usage. Currently it is only possible to see the torrents and the number of requests made for each one. Other statistics dealing with download times or other usage might be helpful.

6.3 Filtering Methods

The current BitTorrent proxy relies solely on blacklists and whitelists to control access to torrents. This approach is simple and provides granular control of torrents, but requires more time be spent by administrators to update the lists. This algorithm could be modified to take advantage of some of the other filtering methods discussed in this paper. Other methods would ease the workload of approving and denying torrents for the administrators.

6.4 Scheduling

The BitTorrent proxy has no method to rearrange the downloads of torrent files based on availability or priority. All torrents are treated equally and downloaded when resources become available. It may be more efficient to schedule downloads based on size and/or the current availability of the torrent. Simply adding the files to a queue may lead to poor performance for torrents that could be considered more important. Future work could be done to improve the scheduling algorithm to account for these factors.

6.5 Seeding

Our proxy requires the administrator to decide which torrents will be seeded. All other torrents are canceled after a configurable amount of time once complete. This behavior has the possibility of seeding several torrents that aren't in demand, while other popular torrents aren't being seeded. It would be beneficial to develop an algorithm to dynamically seed torrents based on demand in the community. If the client could determine which torrents were active, it would be able to use the available bandwidth more efficiently and provide a benefit to more users.

Chapter 7

Conclusion

Our BitTorrent Proxy allows administrators to control the BitTorrent traffic in their networks. It allows organizations to mitigate the risks of inappropriate content and costly bandwidth usage to save money and promote productivity. At the same time, the proxy also benefits users inside and outside of the organization by providing access to torrents with legitimate content.

Bibliography

[1] Daniel Barron. Dansguardian. http://dansguardian.org/?page=introduction.

An introduction to the content filtering software DansGuardian. This software filters Internet content using keywords and phrases, as well as URL matching, PICS, MIME, and extension filtering, as well as POST limiting.

[2] CacheLogic. P2p in 2005. http://www.cachelogic.com/research/2005_slide06.php.

A 2004 study showed BitTorrent was the largest P2P network, a shift towards video sharing, BitTorrent was accounting for as much as 30% of all Internet traffic, and December 2004 saw a crackdown on major BitTorrent sites such as Suprnova.

[3] Adrian Chadd, Robert Collins, Henrik Nordstrom, Alex Rousskov, and Duane Wessels. Squid. http://www.squid-cache.org/.

Squid is an open source web caching proxy additionally providing SSL support, extensive access controls, and full logging. Squid caches web pages and optionally DNS requests as well.

[4] Bram Cohen. Incentives build robustness in bittorrent. http://www.bittorrent.com/bittorrentecon.pdf.

A paper by the designer of BitTorrent that documents the way the client works, and reasons for these decisions. Describes how to publish content, interactions between peers, pipelining, piece selection, and choking algorithms.

[5] Michael Desmond, Liane Cassavoy, Mark Sullivan, and Ramon G. McLeod. Sneaky sharing. *PC World*, 22(10):26–28, October 2004.

Despite a decline in peer-to-peer usage, it still composes a large portion of total Internet bandwidth and thrives in underground circles.

Many users have been scared away, but determined users form exclusive sharing rings, or going offline with cheap, portable mass storage units.

[6] dRD. Mpaa sues six tv bittorrent sites. http://www.afterdawn.com/news/archive/6433.cf.

A May 12, 2005 press release from the MPAA named six BitTorrent sites as targets of a lawsuit. These sites were accused of providing trackers to facilitate the sharing of TV shows.

[7] Leslie Ellis. Bittorrent's swarms have a deadly bite on broadband nets. *Multichannel News*, 27(19):44, May 2006.

Ten BitTorrent users can consume as much as 55% of the upstream bandwidth for a neighborhood DSL line. This causes "best effort" traffic such as email and VOIP services to suffer.

[8] Douglas Ennis, Divyangi Anchan, and Mahmoud Pegah. The front line battle against p2p. In *Proceedings of the 32nd annual ACM SIGUCCS conference on User services*, pages 101–106, 2004.

Describes the reasons that the Ringling School of Art and Design decided to ban peer-to-peer software and the efforts they took to enforce the policy. The authors propose a method to use several open source programs to detect and block peer-to-peer traffic. They also provide the necessary source modifications for those interested in reproducing their filtering system.

[9] Ben Fritz. Antipiracy effort pins tail on edonkey. *Daily Variety*, 135(2):26–27, December 2004.

Describes possible legal implications of peer-to-peer software. Quotes a lawyer stating that a demonstration that a company had knowledge of what they were doing, and substantially aided copyright infringement provides a strong case in line with the lawsuit against Napster.

[10] Eric Krapf. P2p 4 the enterprise? *Business Communications Review*, 35(5):4, May 2005.

Peer-to-peer is more suited for individual rather that business use. Many businesses have tried to keep peer-to-peer bandwidth off corporate networks because of the large bandwidth usage.

[11] Sai Ho Kwok. File-sharing activities over bt networks: Pirated movies. *ACM Computers in Entertainment*, 2(1):Article 01, April 2004.

A one-month study of the requests and postings of the Lord of the Rings series near the release of the final movie. Several forums were monitored for activity regarding the movie, and the posted files were downloaded to verify that they were actually the requested movie. Several patterns seemed to emerge including increased demand for other movies from the series, higher volume near the release date, and an increase over holidays.

[12] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

Proposes a method of spam classification based on the use of Bayesian Classifiers. Mail messages are represented as feature vectors. Each dimension of the vector space is defined as a given word in the corpus of all the messages seen. Each individual message is represented as a binary vector listing which words are in the message. A probabilistic classifier can be learned using training messages to detect unwanted mail.

[13] Richard H. Stern. Vicarious liability for infringement. *IEEE Micro*, 24(5):6, 2004.

Discusses the question of when it is proper to hold one party liable for the actions of another. Generally, courts have required that there is an intention to promote infringing behavior with copyright laws. Also argues against a House Bill being debated in Congress as being too broad.

[14] Tao Yang. Pornseer. http://www.yangsky.us/products/porndetect/index.htm.

PNWatch is a product that uses image recognition technology to search for images of human anatomy and sexual positions. This product is part of an open source project PornSeer and a commercial product PornSeer Pro.

Appendix A

Installation

This appendix covers the basic installation and configuration steps for our BitTorrent Proxy.

A.1 BitTorrent Proxy

In order to function properly, the BitTorrent Proxy requires an installed BitTorrent client, a MySQL database, and the Python scripting language.

All of the files in the btProxy directory should be copied to the server. The btProxy.conf file contains all the configuration settings and needs to be modified before running the proxy. Once the configuration changes have been made, the proxy can be started by using the command ‘python btProxy.py’.

A.1.1 Required Configuration

The btProxy.conf configuration file contains several required parameters. In order for the proxy to contact the database, the databaseLocation and databaseName parameters must be set. These parameters simply list the database server location and the name of the database for the Python MySQL libraries. In addition to the location, the databaseUser and databasePassword parameters must be provided for a user with access to update the database listed in databaseName.

The fileRepository location must also be specified before launching the proxy. The BitTorrent client uses this location to store downloaded torrents. Also, to enable the users to download these files, the repositoryURL needs to be specified. This location will be prepended to the downloaded file names to provide the download link. This URL should have the proper permissions to allow users to download the completed torrents.

A.1.2 Optional Parameters

The btProxy.conf can also be used to specify the maximum bandwidth to use for uploading and downloading torrents. The parameters maxUpBandwidth and maxDownBandwidth control the maximum bandwidth in Kbps to use for uploading and downloading torrents respectively.

By default, the btProxy will log information to btProxy.log in the btProxy directory. If the log should be located somewhere else, it can be changed using the logFile parameter. The file will be automatically created, but this location should contain an existing directory.

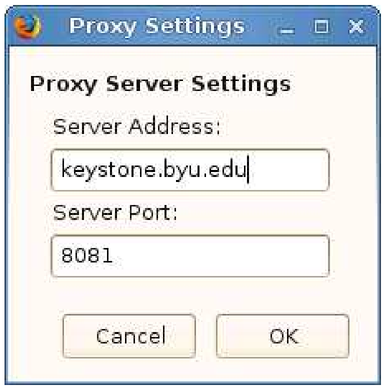
A.2 Web Administration

To install of the Web Administration component, copy all the files from the btpWeb directory to an Apache/PHP webserver and modify the db.inc file. The parameters at the top of the db.inc file should provide a valid host, database, username and password for access to a database with the users table.

Care should be taken to insure that the db.inc file will not be served by the web server. If users obtain the username and password from the db.inc, they will have administrative access.

A.3 Firefox Extension

In order to install the Firefox extension, just open the btp.xpi file in Firefox. The extension will be installed, and you will be asked to restart the browser. After the restart, access the Tools menu and select the BitTorrent Proxy Settings option. The window shown in Figure A.1 will be displayed. Enter the name or IP address of the BitTorrent Proxy server in the Server Address box, and the port in the Server Port box.


Proxy Settings

Proxy Server Settings

Server Address: keystone.byu.edu

Server Port: 8081

Figure A.1: BitTorrent Proxy Settings

Appendix B

Database Tables

The following is a list of the database tables currently in use for the BitTorrent Proxy.

B.1 Config

The config table contains information about the current configuration of the BitTorrent client.

Field	Type	Null	Key
fileRepository	char(255)	Yes	
repositoryURL	char(255)	Yes	
maxUploadBandwidth	int(11)	Yes	
maxDownloadBandwidth	int(11)	No	

B.2 Users

The users table contains information about all users that have registered with the proxy.

Field	Type	Null	Key
firstName	varchar(50)	Yes	
lastName	varchar(50)	Yes	
email	varchar(50)	Yes	
username	varchar(50)	No	Primary
password	varchar(32)	No	
description	varchar(255)	Yes	

B.3 Whitelist

The whitelist table contains all URLs which have been approved safe by an administrator.

Field	Type	Null	Key
url	varchar(300)	Yes	

B.4 Torrents

The torrents table contains all requested torrents along with information about those torrents.

Field	Type	Null	Key
file	varchar(100)	No	Primary
url	varchar(200)	No	
download	varchar(300)	Yes	
seed	tinyint(4)	Yes	
approval	tinyint(4)	Yes	
complete	tinyint(4)	Yes	
requests	bigint(20)	Yes	
size	bigint(20)	Yes	
user	varchar(50)	Yes	Foreign
description	varchar(255)	Yes	